uc3m | Universidad Carlos III de Madrid

University Degree in Computer Science and Engineering
2017/2018

Bachelor Thesis

"Use of IBM Watson technology for cognitive analysis"

Jerónimo Javier Salcedo Marangón

Tutor
José Antonio Iglesias Martínez

Leganés, September 2018

CONTENTS

LIST OF TABLES

LIST OF FIGURES

AKNOWLEDGEMENTS

First of all I would like to thank my family for supporting me all of these university years, with special mention to my sisters. Also I would like to thank all those colleagues and friends I've made over the years in the university that were always there when I needed them. I would like to thank all those teachers that teach their subjects with passion and try to make their classes as appealing and comfortable as possible, making a better and easier learning experience for us students. Lastly I would like to thank my tutor José Antonio Iglesias for helping and guiding me through this project as well as answering all my doubts even when he was on vacations.

ABSTRACT

This document presents the design, development and evaluation of a chatbot using IBM Watson services.

The purpose of this bot is to create an assistant that uses several services to aid the user whenever the user needs help creating a CV. This bot will be a prototype due to the costs of the services, and the functionalities will be limited by the service type. However, how this chatbot has been designed and created is a very relevant task.

The main idea is to create a chatbot that helps the user with little experience create a CV/Resume with somewhat a human-like (as much as possible) guidance through all the process that makes it easier for a user that has never made a CV to create their first one. It may also help other users that might want guidance or might be missing something from their CV.

This project has been developed with IBM Watson services, as well as nodejs and npm. HTML and javascript languages have been used to handle the chatbot interface and connections between services.

ACRONYMS AND DEFINITIONS

ACRONYMS

- NLP: Natural Language Processing

- NLU: Natural Language Understanding

- AI: Artificial Intelligence

- AIML: Artificial Intelligence Mark-up Language

- NPM: Node Package Manager

- GDPR: General Data Protection Regulation

TERMINOLOGY DEFINITION

- Utterance: something the user says.

- Intent: this is the meaning or objective of the user within a given message input.

- Entity: relevant class of object for the user's purpose. It may be a person, company, number, pattern, date...

- Context variable: a variable that Watson Assistant uses to analyze the context of the conversation.

- Dialog: it is the sum of all nodes that create the conversation flow, defining the conversation the chatbot might have.

CHAPTER 1

1.- INTRODUCTION

This chapter is an introduction to the project, where the objectives and the motivation behind the project is stated.

1.1.- Motivation

Since some time now, the demand for chatbots has been growing, and at the same time, the technologies that chatbots use, or that can be used along with chatbots have been improving as well, such as AI (Artificial Intelligence) and NLP (Natural Language Processing) This, combined with its ease of use and the guidance that they provide, makes chatbots a very appealing option for many companies when it comes to customer service. There are a lot of use cases for which chatbots can be used and they can be combined with several services to achieve different objectives.

Also, many people finishing their career need to make their first CV and some are clueless on where to start. An application to help them out with that task sounds good.

Considering this and the accessibility to IBM Watson dedicated services that revolve around chatbots, the creation of a prototype chatbot to help people create their first CV seems like a good idea.

1.2.- Project objectives

The main objective of this bot is to be able to guide a user when creating a CV. This tool is aimed at those users who have never created a CV before, although it can be useful to other users that have already created a CV before but want tips or need help with a particular area of it.

There may be problems like knowing which CV template to follow or how to distribute information introduced into the CV. Other difficulties may arise, such as the connections between services and how they interact with each other.

In order to achieve this objective, several smaller objectives will be followed:

- Creation of the conversational flow for the chatbot
- Creation of an interface for the chatbot
- Establish connection between the app and the chatbot service

- Creation of a structure that sends information from the system to the Watson Assistant service and receives it, then processes it and gives an answer to the user.

Once all these steps have been completed the final objective may be achieved by adding all the previous steps into one.

1.3.- Limitations

This project is limited by the IBM services packs available and will not be as functional as it could be with a premium version of said services. Considering this, the project could be thought of as a prototype of a chatbot to help the user create the first CV and serve as guidance on how to start looking for places to work in and where to send this first CV.

1.4.- Structure of the document

The structure of the document has six chapters and 3 appendices with the following purpose:

1. Introduction: This chapter will explain the motivation for this project, as well as the objective that wants to fulfil.

2. State of the art: This chapter explains what the current situation around chatbots is, and lists some chatbots and tools.

3. Development environment: This chapter contains information about the environment in which the project was developed as well as a brief explanation of the tools used.

4. Proposal: This chapter defines the use cases and requirements as well as exposing the design behind the project.

5. Evaluation: This chapter will show what has been done to evaluate the project to check it is working correctly.

6. Legal framework and Socio-economic impact: This chapter explains the legal restraints that may be applied to the project as well as which might be the socio-economic impact of itself. It also contains the budget for the project.

7. Conclusions and future work: This final chapter reflects the conclusions withdrawn from the development of the project and future work that may arise after finishing the project.

- User manual: This appendix explains how to run the bot and any installations or configurations needed for it.

CHAPTER 2

2.- STATE OF THE ART

This chapter lays out the current state of chatbots, some of their many uses and what needs or should be improved. It also lists some of the many chatbots that are available and some tools to create chatbots.

2.1.- Chatbots

A chatbot is a tool used for an specific purpose within a set context. Normally, it is easy to build and does not take much time (depends on the objective of the chatbot and the complexity wanted for it). Even though they might seem similar to personal assistants, they are quite different, since personal assistants are not bound to a specific area or context and can handle many of the tasks the user needs, such as making appointments, telling the user what weather will be like for any day or calling someone, and takes a long time to build. Unlike personal assistants, a chatbot can only handle tasks within a given context.

Chatbots are currently used for specific tasks within a context range, such as selling a product, answering questions related to something specific, helping a customer with a task or just having a normal conversation (but nothing too complex). As chatbots have different objectives, they also have different focuses on how they want to perform, for example, chatbots made for selling products focus on trying to give information that the customer finds appealing and easy to understand and use, while chatbots that are made for chatting focus on being as human as possible to keep the conversation going.

The current problem with chatbots is for the chatbot to be able to correctly understand what the user intent is given a set context, and being able to reply back to the user in a rather human way with the correct information. This means, that the chatbot relies on natural language understanding techniques to be able to work properly, and will perform better as those techniques improve. Also, many of the chatbots have predefined conversations in which if the user steps slightly out of context or uses any expresion, the chatbot will not be able to work properly as it has a set question-answer path. A possible answer to this could be implementing better NLU and NLP, as well as a parser, for example, to separate sentences and be able to analyze the parts of it to be able to understand the context correctly. Also other AI techniques can be used to give the bot a wider variety of answers depending on the context or if the user steps out of context, in order to keep the flow of the conversation stable. Some chatbots have some of these features, but there is much more to be achieved.

2.2.- Current Chatbots

This section will list some of the current chatbots available and what they are used for:

2.2.1.- ELIZA

ELIZA is considered the first chatbot. Created by Joseph Weizenbaum in 1966, ELIZA was meant to be a parody of the questions rogerian psychotherapists make. At first it managed to fool people into thinking it was a human person they were talking to, but as the conversation went on, it started turning incoherent. Weizenbaum made the bot so that it recognized keywords and asked about them, understanding its context and trying to create an adequate answer to what the user said. This would be the base for other chatbots. [1]

2.2.2.- CV/Resume chatbot

This bot has a similar idea to the project one, but works the opposite way. Instead of creating a CV you create a chatbot. As there are many of this bot (one for each CV), this bot is not a concrete bot with a name, but the idea of turning your CV/resume into a chatbot that has all the information concerning your CV/resume and talks about it with the user. There are several tools that can be found online that are available for this purpose and also have some guidance through the process.

2.2.3.- Imsomnobot

Imsomnobot is a chatbot designed for people that have sleeping problems and are up at late hours. It helps them get distracted and its solely function is to chat with the user. [2]

2.2.4.- Poncho

Poncho was a chatbot for weather forecast. It would show the weather anytime, anywhere and it also had weather related jokes. Now it has been transformed into a beverage selling bot, but can still be talked to at Facebook Messenger before it gets shut down. [3]

2.2.5.- Melody

Melody is a chatbot inside of "Baidu Doctor" application. It is a chatbot which's objective is to gather information from patients' symptoms to make it easier for the patient to get medical help, or to have extra information from the patient for the doctors to make a more accurate treatment. [4]

2.2.6.- RightClick

RightClick.io is a chatbot that uses AI to help the user build a fully functional website from zero. It gives the user templates and a lot of facilities for the website. When trying to change topics or if you try to leave the context it was created for, it tries to lead you back on track, instead of answering "I do not understand you." like many other bots do. [5]

2.2.7.- Mitsuku

Mitsuku is a chatbot created by Steve Worswick in 2005. Created with AIML* technology, this chatbot is a three-time Loebner Prize winner and is claimed to be the most human-like chatbot there is right now. Its aim is to appear as human as possible. It is based on Alice, a natural language processing chatbot, and uses its libraries as a base. [6]

2.3.- Chatbot creation tools

This section will show some of the tools used to create chatbots:

2.3.1.- ChatFuel

ChatFuel is a service created with the objective of making easier the task of building bots in 2015. With a start on telegram, this service eventually moved to Facebook Messenger, and currently focuses on chatbot creation for said messenger.

2.3.2.- Bottr

Bottr is a framework for chatbot creation. It is built on top of node.js and is normally deployed using Heroku cloud platform. It is also focused on chatbot creation for Facebook Messenger.

2.3.3.- Motion.ai

Motion AI is a company from Chicago that created a bot-builder app that guides you through all the steps of building the chatbot. It also can be deployed to several platforms. It is said to be characterized by its bot “modules” which hold the logic to several bot features.

2.3.4.- FlowXO

FlowXO is a platform for chatbot creation and hosting. These chatbots are meant for messaging apps, and gather information from different sources such as users, google sheets or trello, to create and improve answers from the chatbot.

2.3.5.- Dialogflow

Dialog flow is a platform for conversation-flow building created by Speaktoit, which was bought in 2016 by Google. It is powered by Google's machine learning and runs in Google's cloud environment. It is able to be deployed on any platform and many smart devices.

2.3.6.- IBM Watson Assistant

Watson Assistant is the tool that will be used for this project. It is integrated in the cloud services provided by IBM. It uses Watson AI and NLU to build the conversation and understand intents and entities. It also has a GUI for the user to ease the process of creating the conversation flow, intents and entities.

CHAPTER 3

3.- DEVELOPMENT ENVIRONMENT

This chapter shows a general view of the tools used to develop the project and how they work.

3.1.- Conversation flow tools

The conversation flow is the core of the chatbot that handles all answers provided by the bot. It uses Watson AI and NLU to understand users intents and entities, as well as storing context variables that help with the flow of the conversation.

3.1.1.- Watson Assistant

Watson Assistant is a service provided by IBM that is within IBM Cloud and is accessible to anyone with an IBM ID. Although the code behind all the functionalities it has is private, the tool gives the ability to modify parts of the code from the chatbot and the conversation flow to personalize it and mold it to fit whatever objective it needs to fulfil. It also has a GUI to make it easier for users to create conversation flows without the need of knowing how to write code. Even though creating a simple bot is rather intuitive, and some may say easy, as more functionalities want to be added to the bot the complexity and difficulty starts to grow bigger, but the rate at which it grows is reduced by its simple GUI and the guidance it provides. This GUI can only be used for creating the conversation flow and testing it. If the bot needs to be implemented inside an app it needs new code to hold it and its functionalities. As of now it is still being developed and new functionalities are added periodically. It has 3 packages from which to select: lite(free), standard and premium. Each one has its limitations, but the premium one has extra functionalities (like clearing ambiguity) and can be personalized to fit the customer's needs. Overall it is an easy and intuitive tool to use that eases the process of creating a chatbot.

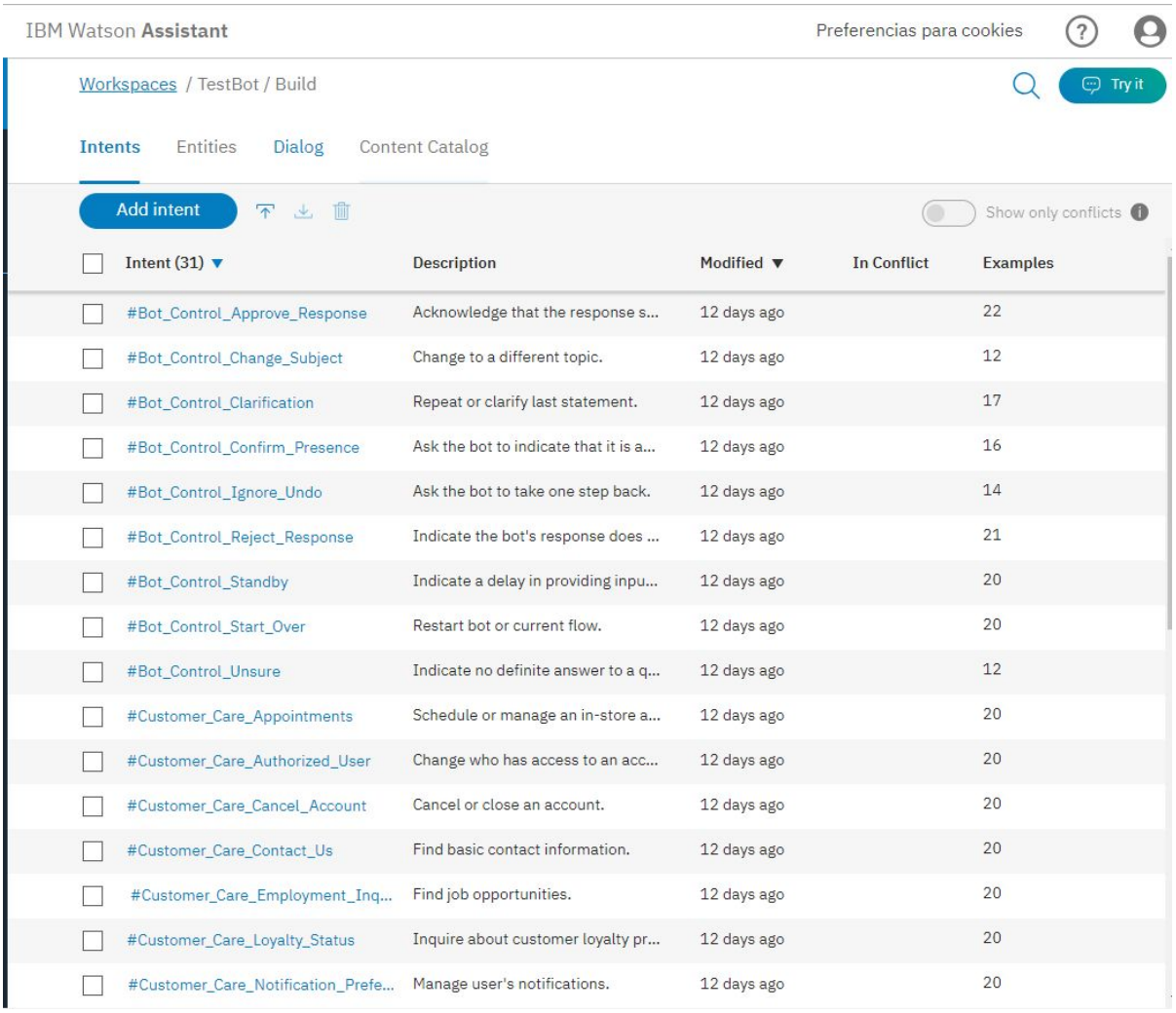

Figure 3.1: Watson Assistant Intents overview.

Figure 3.1 displays an overview of the intent panel of the tool. In this section the input from the user that can be understood by the bot is listed with severals examples for training.

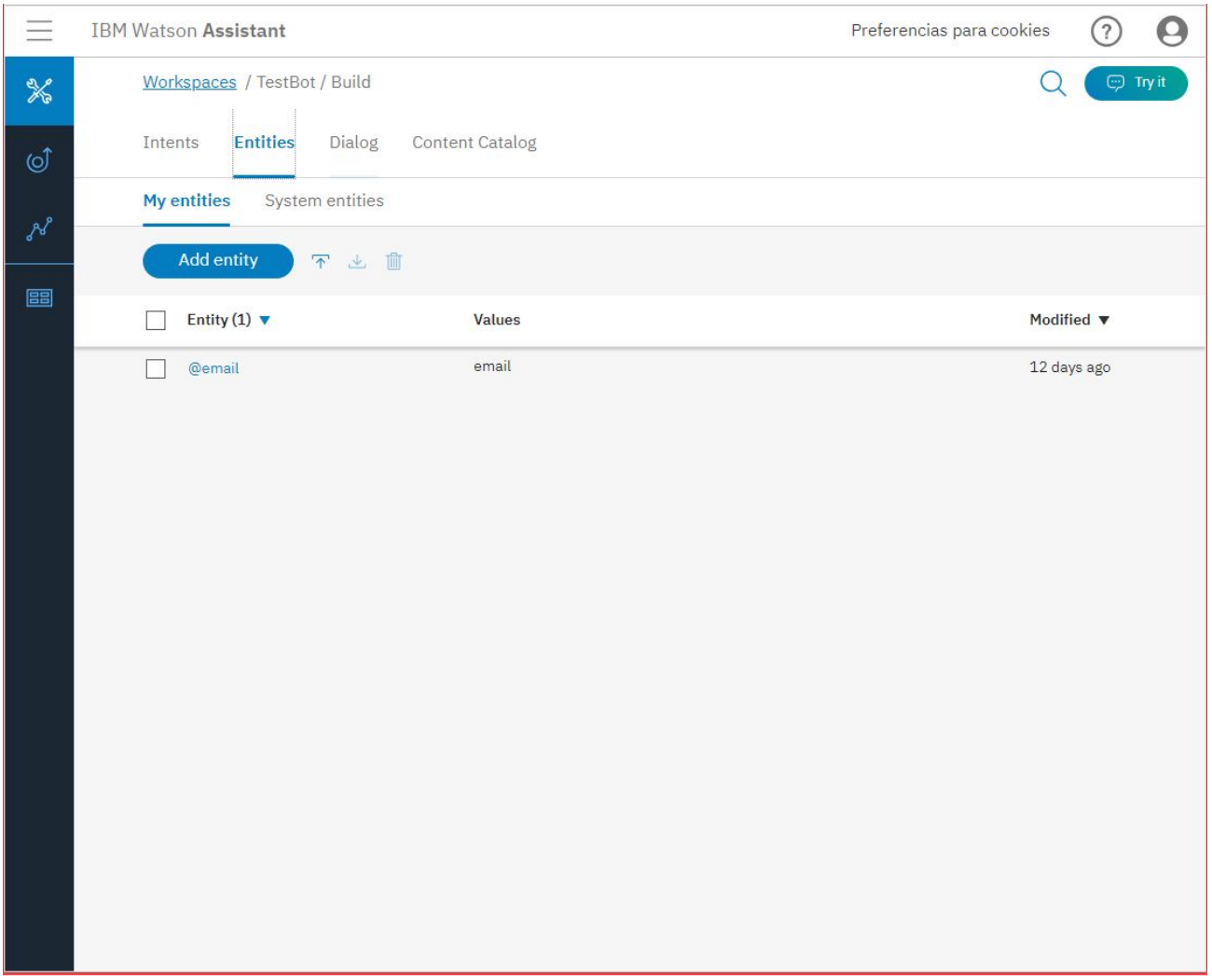

Figure 3.2: Watson Assistant entities overview.

In figure 3.2 an overview of the entity panel can be seen. Here entities the bot can recognize are stated. The tool brings basic entities that can be included such as numbers, dates or names.

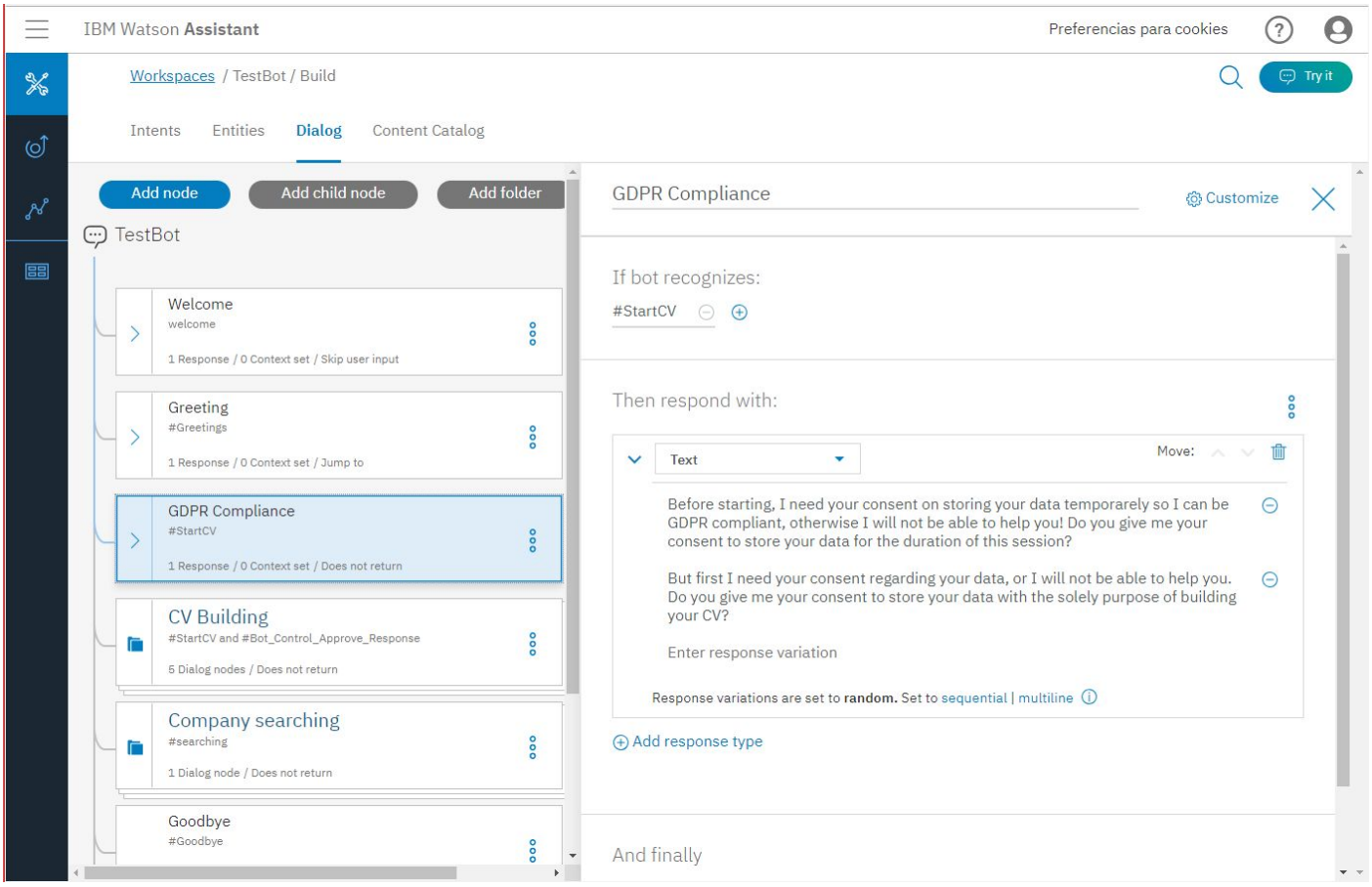

Figure 3.3: Watson Assistant conversation flow overview.

This figure shows the conversation flow. It is composed of nodes which recognize user input and give an answer according to it.

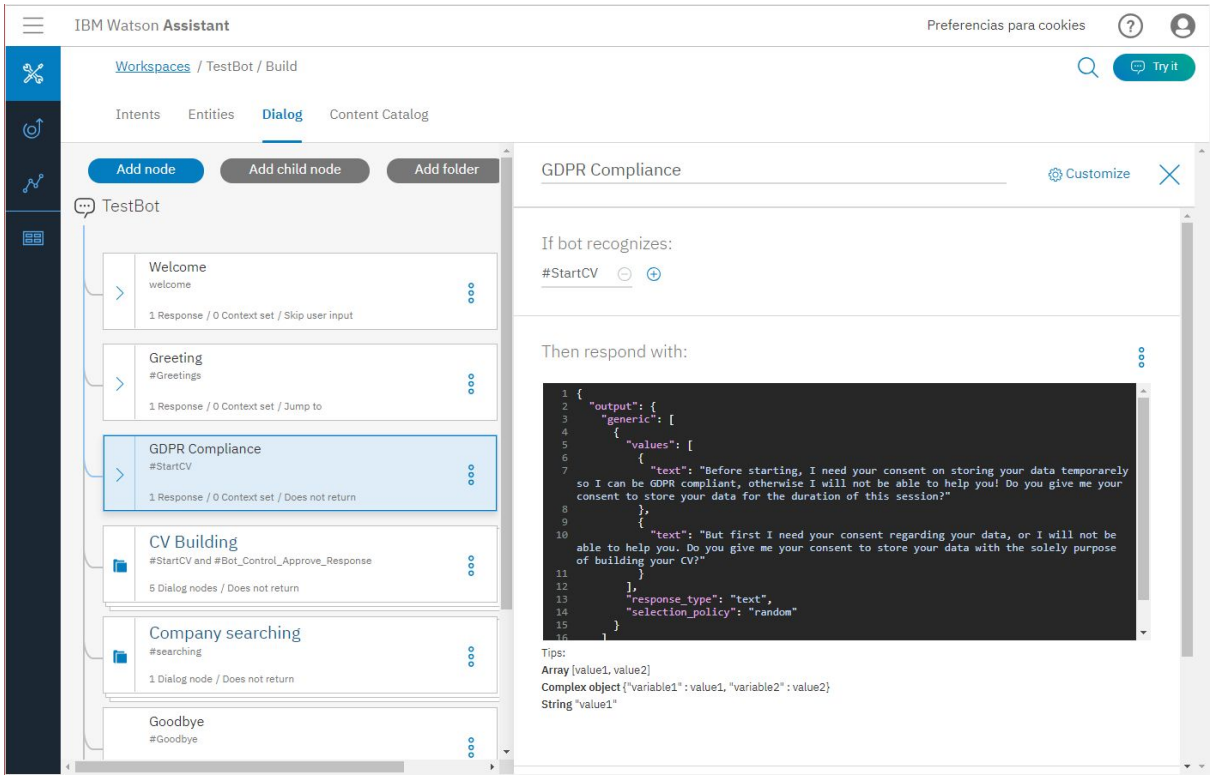

Figure 3.4: Watson Assistant JSON node overview.

Each node's code can be accessed to be personalized further with additional functionalities that may not come with the tool. The code of each node is a JSON file.

3.1.2.- IBM Cloud

IBM Cloud is an IBM platform that holds a wide variety of services that range from storing data to hosting applications and services that these applications may use. All data stored inside it is by default encrypted. This platform holds both the conversation flow logic and application that runs the bot interface. It has limitations and if more storage capability is needed (or wanted) it will come with a set price for it. It also has a catalog with all the services it provides as well as information and guides. IBM Cloud also provides a command line interface to interact with it and upload data and applications as well as downloading and modifying existing that that is within the cloud.

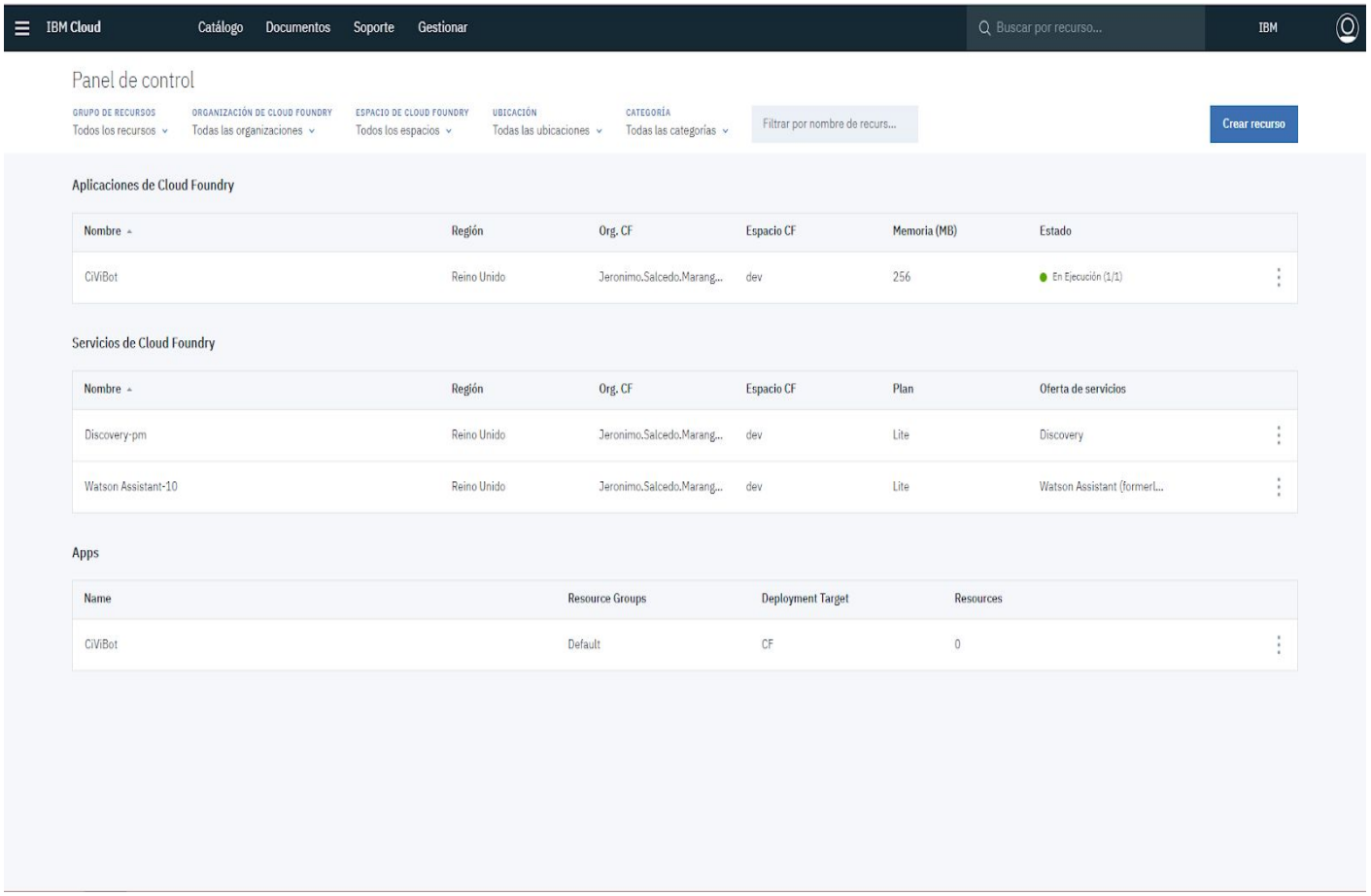

Figure 3.5 : Overview of IBM Cloud control panel.

In the figure 3. an overview of the interface of IBM Cloud control panel can be seen. It shows a list of the apps that are running (first list item) the services that the account has in use and the apps that have been uploaded. From this control panel all services and apps can be accessed and managed by clicking on them.

3.2.- App deployment environment

The chatbot is deployed into an application built with Node.js. It uses several modules that can be found in the "*package.json*" file and can be installed easily by using the command "*npm install*" in the command line window. Whilst the conversation flow functionality is built with the IBM Watson services, all the interface related components are built using Node.js and its module system. All the script part is built with javascript, style using css and frontend using HTML.

3.2.1.- Node.js

Node.js is a JavaScript runtime environment which has the JavaScript engine V8 from chrome. It uses a package system with a package manager called NPM which is able to implement libraries into the project as modules. It was devised to run standalone javascript applications. It has an event-driven non blocking I/O model that can also handle HTTP requests. [7][8]

3.2.2.- NPM

NPM is the package manager for Node.js which is able to import libraries into any Node.js project. These packages come in the form of modules and do not impact existing code. It can also be used to import your own files into the project. To do this it uses a function called "*require*" which needs a path for the file to be required (or the name of the module) and returns a *module.exports* which is already implemented in imported modules but has to be defined in own files.

CHAPTER 4

4.- PROPOSAL

This chapter describes the project development that has been carried out.

4.1.- Work Methodology

This section will expose the work methodology that has been carried out as well as the phases of development the project has gone through.

The project follows an incremental model which is similar to a "*multi-waterfall-like*" cycle. Like in the waterfall model, the project goes through requirements, analysis, design, implementation and testing phases, but it is repeated for every module that adds functionality to the project as a whole. This way, the projects works by incrementally adding functionality in the form of modules that are added to the previous ones that have already been finished. The requirements phase is the same for all the modules, but for each module a part of the requirements are analysed, designed, implemented and tested.

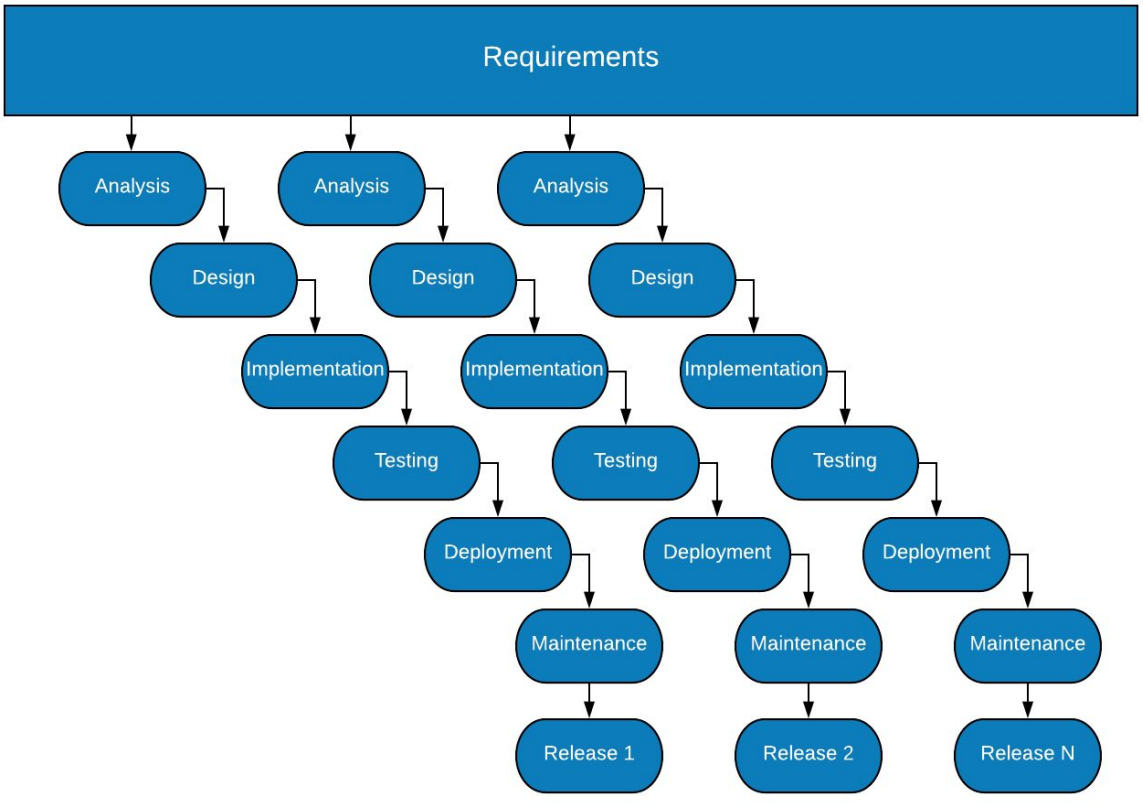

Figure 4.1: Incremental development model.

These are the different phases:

- Requirements: In this phase, the requirements for the project to fulfil are defined clearly.

- Analysis: This phase analyses the tools and resources that are going to be used in the project to gain as much productivity and efficiency as possible from them.

- Design: In this phase the components of the module are designed taking into consideration part of the requirements stated before.

- Implementation: This phase's objective is to implement the design from the previous phase.

- Testing: This phase implements the module implementation of the previous phase into the existing project and tests it to make sure everything works correctly.

- Deployment: In this phase, the new version with the inserted module is deployed into the existing production system.

- Maintenance: This phase supports the current version of the project until a new version is released.

- Release: The current functional version of the product is released for use.

4.2.- Use Case Analysis

This part will show the different use cases that are covered by this project.

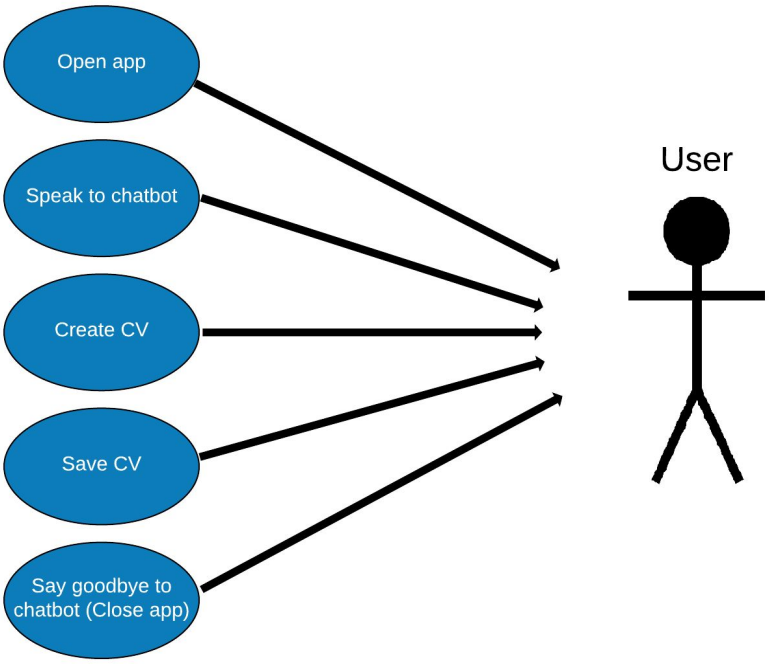

Figure 4.2: Use case user diagram.

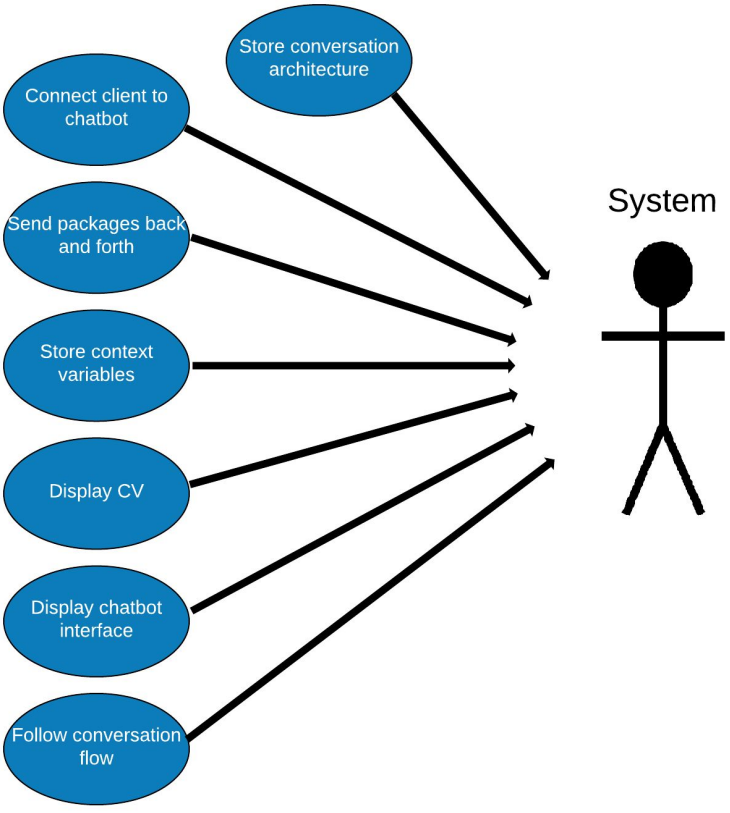

Figure 4.3: Use case system diagram.

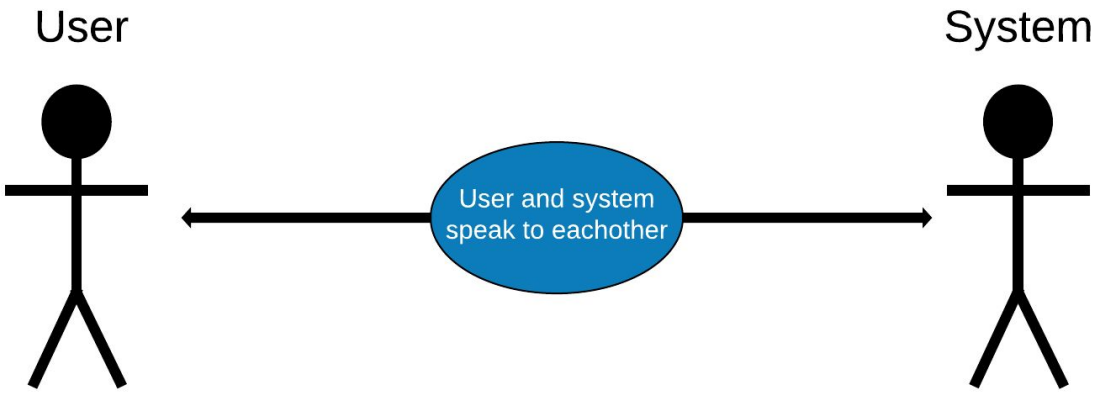

Figure 4.4: Use case user-system diagram.

The use cases will now be defined in the following table template:

Use Case	
ID	CBU-XY
Name	Name of the case
Actors	Actors involved
Description	Description of the case
Effects	Effect of the case

Table 4.1: Use Case template.

The definition of each field is listed below:

- ID: This field contains a unique identifier for each use case.
- Name: Contains the name of the use case.
- Actors: Reflects the entities that take part in the use case.
- Description: Brief description of the use case stating a scenario.
- Effects: An effect that happens when the given use case is performed.

Use Case	
ID	CBU-01
Name	Open app
Actors	User
Description	The user uses the link provided to access the application.
Effects	The user is now connected to the application which holds the chatbot.

Table 4.2: Use Case 01.

Use Case	
ID	CBU-02
Name	Speak to chatbot
Actors	User
Description	The user engages in conversation with the chatbot and starts sending messages through the app interface.
Effects	The user has sent a package to the chatbot and now awaits for a response.

Table 4.3: Use Case 02.

Use Case	
ID	CBU-03
Name	Create CV
Actors	User
Description	Through the conversation with the chatbot, the user provides information to build a CV.
Effects	The CV is built with the information provided by the user.

Table 4.4: Use Case 03.

Use Case	
ID	CBU-04
Name	Save CV
Actors	User
Description	Once the CV is complete, the user can save a copy of the CV provided by the system.
Effects	A CV copy is saved in the user's computer.

Table 4.5: Use Case 04.

Use Case	
ID	CBU-05
Name	Say goodbye to chatbot (Close app)
Actors	User
Description	The user has finished using the chatbot and ends the conversation.
Effects	The chatbot conversation has ended.

Table 4.6: Use Case 05.

Use Case	
ID	CBU-06
Name	Store conversation architecture
Actors	System
Description	The system stores the architecture the conversation must follow.
Effects	The chatbot now has a conversation architecture.

Table 4.7: Use Case 06.

Use Case	
ID	CBU-07
Name	Connect client to chatbot
Actors	System
Description	When the system starts running, it creates a connection between the client and the chatbot.
Effects	A connection is created between the client and the chatbot.

Table 4.8: Use Case 07.

Use Case	
ID	CBU-08
Name	Send packages back and forth
Actors	System
Description	Through the client-chatbot connection, the system sends packages with the input from the user to the chatbot and the response from the chatbot to the user.
Effects	Packages arrive at their destination with the information provided by both sides.

Table 4.9: Use Case 08.

Use Case	
ID	CBU-09
Name	Store context variable
Actors	System
Description	The system stores context variables collected from the user's input that will be after sent to the chatbot.
Effects	A context variable is stored within the system.

Table 4.10: Use Case 09.

Use Case	
ID	CBU-10
Name	Display CV
Actors	System
Description	The system shows the information collected in the CV from the user to the user so it can be saved into the user's computer.
Effects	The CV information is shown to the user.

Table 4.11: Use Case 10.

Use Case	
ID	CBU-11
Name	Display chatbot interface
Actors	System
Description	The system display a UI in which the user can send messages to the chatbot and viceversa.
Effects	An UI is displayed to the user.

Table 4.12: Use Case 11.

Use Case	
ID	CBU-12
Name	Follow conversation flow.
Actors	System
Description	The system guides the user through the conversation according to the provided conversation architecture for the chatbot.
Effects	The user is guided through the conversation.

Table 4.13: Use Case 12.

Use Case	
ID	CBU-13
Name	User and system speak to each other
Actors	User, System
Description	Through the UI for the chatbot provided before, the user types messages to the chatbot and the chatbot sends messages back to engage in a conversation.
Effects	A conversation between the user and the chatbot is being held.

Table 4.14: Use Case 13.

4.3.- Requirements Definition

This sections shows the requirements that the project needs to fulfill, from which part of them will be selected for each module to be implemented. There will be user requirements, functional requirements and non-functional requirements.

The definition of the requirements will be shown in the form of the following table template:

Requirement	
ID	UR-XY/FR-XY/NFR-XY
Name	Name of the requirement
Description	Description of the requirement
Relation	Relation with other requirements
Priority	Priority of the requirement

Table 4.15: Requirement table template.

Each field is defined as the following:

- ID: Unique ID given to the requirement.
- Name: Name given to the requirement.
- Description: Brief description of the requirement.
- Relation: Relation or pre-requirements the requirement may have,
- Priority: It shows the requirements importance within the system. There are 3 priorities:

  · High: The requirement is needed to be implemented within the first modules released in order to have critical functionality.

  · Medium: The requirement adds functionality but will be implemented after the high priority ones.

  · Low: The requirement is not part of the core functionality of the system and does not need to be implemented as early as the rest.

Requirement	
ID	UR-01
Name	Connection
Description	The user must be able to connect to the app via the url.
Relation	FR-01
Priority	High

Table 4.16: Requirement table 01.

Requirement	
ID	UR-02
Name	Chatbot access
Description	The user must be able to access the chatbot interface.
Relation	FR-02
Priority	High

Table 4.17: Requirement table 02.

Requirement	
ID	UR-03
Name	Conversation
Description	The user must be able to talk with the chatbot.
Relation	FR-03
Priority	High

Table 4.18: Requirement table 03.

Requirement	
ID	UR-04
Name	CV building
Description	The user must be able to build a CV while talking to the chatbot.
Relation	UR-03, FR-04
Priority	Medium

Table 4.19: Requirement table 04.

Requirement	
ID	UR-05
Name	Saving CV
Description	The user must be able to save the CV created.
Relation	UR-03, UR-04, FR-05
Priority	Medium

Table 4.20: Requirement table 05.

Requirement	
ID	UR-06
Name	End conversation
Description	The user must be able to end the session with the chatbot.
Relation	UR-03, FR-06
Priority	Low

Table 4.21: Requirement table 06.

Requirement	
ID	FR-01
Name	App hosting
Description	The app must be hosted with a url to link to it for the user to access the app.
Relation	UR-01
Priority	High

Table 4.22: Requirement table 07.

Requirement	
ID	FR-02
Name	App interface
Description	The system must have an interface to navigate and access the chatbot.
Relation	UR-02
Priority	High

Table 4.23: Requirement table 08.

Requirement	
ID	FR-03
Name	Chatbot interface
Description	The system must have an interface that holds the chatbot itself.
Relation	UR-03
Priority	High

Table 4.24: Requirement table 09.

Requirement	
ID	FR-04
Name	Chatbot hosting
Description	The system must be connected to the Watson Assistant workspace in IBM Cloud.
Relation	UR-04, FR-01, FR-03
Priority	High

Table 4.25: Requirement table 10.

Requirement	
ID	FR-05
Name	Conversation flow
Description	The system must be able to follow the conversation flow contained in the Watson Assistant service.
Relation	UR-04, UR-05, FR-03, FR-04
Priority	Medium

Table 4.26: Requirement table 11.

Requirement	
ID	FR-06
Name	Show CV
Description	The system must show the CV to the user once completed in order to be saved by the user.
Relation	UR-05, FR-05
Priority	Medium

Table 4.27: Requirement table 12.

Requirement	
ID	FR-07
Name	End connection
Description	The system must end the connection once the user finishes chatting.
Relation	UR-06
Priority	Low

Table 4.28: Requirement table 13.

Requirement	
ID	FR-08
Name	Context variables
Description	The system must be able to store and send context variables to the Watson Assistant service.
Relation	FR-04
Priority	High

Table 4.29: Requirement table 14.

Requirement	
ID	NFR-01
Name	Use of HTML, CSS and JS
Description	The project must be coded using HTML, CSS for styling, and JavaScript.
Relation	-
Priority	High

Table 4.30: Requirement table 15.

4.4.- Project Planning

The following phases have been planned for the project:

1. Specify the idea to be developed.

2. Investigate available tools to develop the idea.

    - Watson Assistant
    - Watson Discovery
    - IBM Cloud
    - Node.js
    - NPM

3. Define use cases and requirements.

    - Use cases
    - Requirements

4. Design of the system.

    - Architecture design
    - Conversation flow design

5. Implementation of the design.

    - Conversation flow implementation
    - Interface implementation

6. Testing of the design.

    - Connection test
    - Conversation test

7. Creation of the report.

This project started around the 15th of April (the start was delayed due to some administration problems) and ended the 2Xth of September, adding up to a total of about 5 and a half months. Also, the development of the project was slowed down due to other tasks such as work, formation courses and university lab activities. Something else to take into consideration is the fact that whilst investigating the tools to be used in

the project development, the main idea varied due to limitations of the tools or flaws in how to implement idea that were not compatible with the way the tools worked.

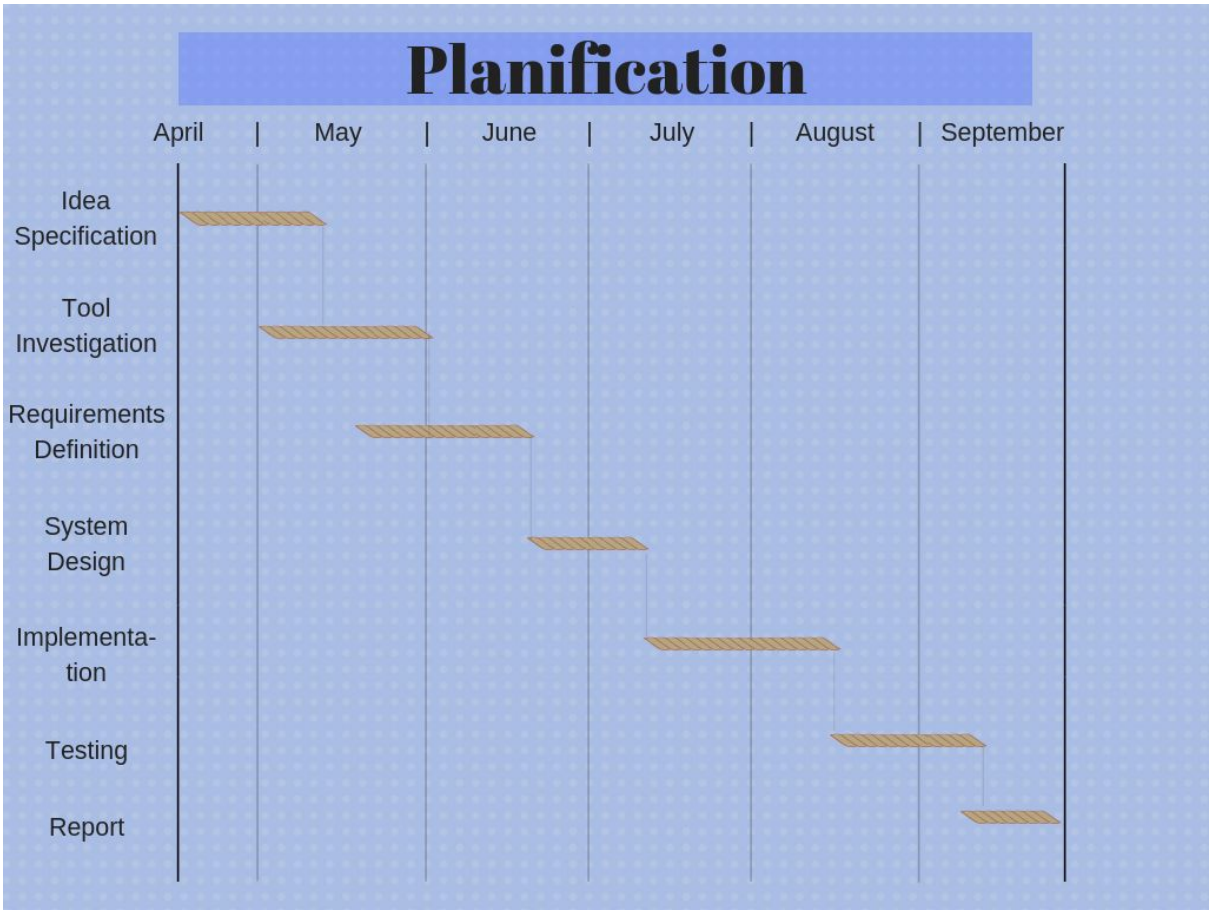

Figure 4.5: Gantt diagram. Planification.

4.5.- Design

This section will contain the most relevant design parts, such as the architecture design and the chatbot's conversation flow design, as well as the flow diagram for the app in general..

4.5.1.- Architecture

For the general architecture of the system this figure will show an overview:

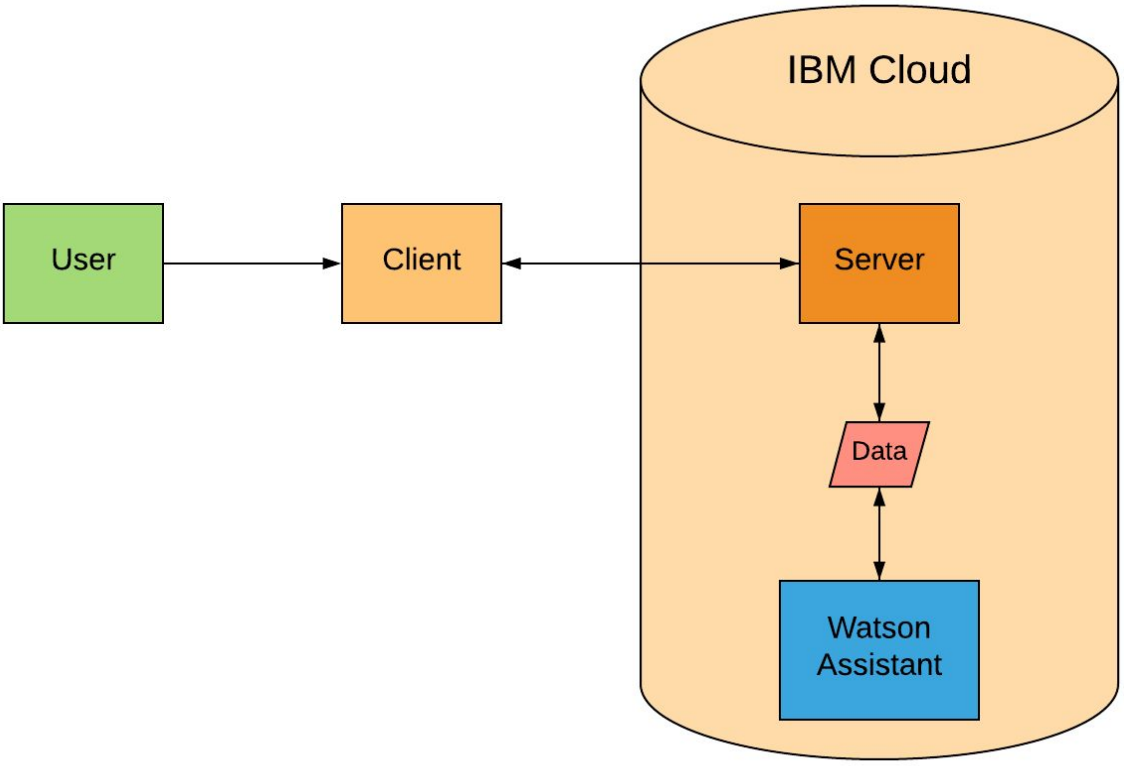

Figure 4.6: General architecture.

The user will connect to the client via the url provided and will interact with it. When a message is typed by the user, the client sends it to the server where the package with the data needed is prepared and sent to the Watson Assistant API via an api call. Then Watson Assistant processes the call and sends back an answer to the server, which is then processed and shown to the user via the client interface.

The client interface is coded in HTML and JavaScript, using CSS for styling and JavaScript and Jquery for some animations and effects. The server side is coded in JavaScript.

4.5.2.- Diagram flow

This figure will help with the description of the conversation flow:

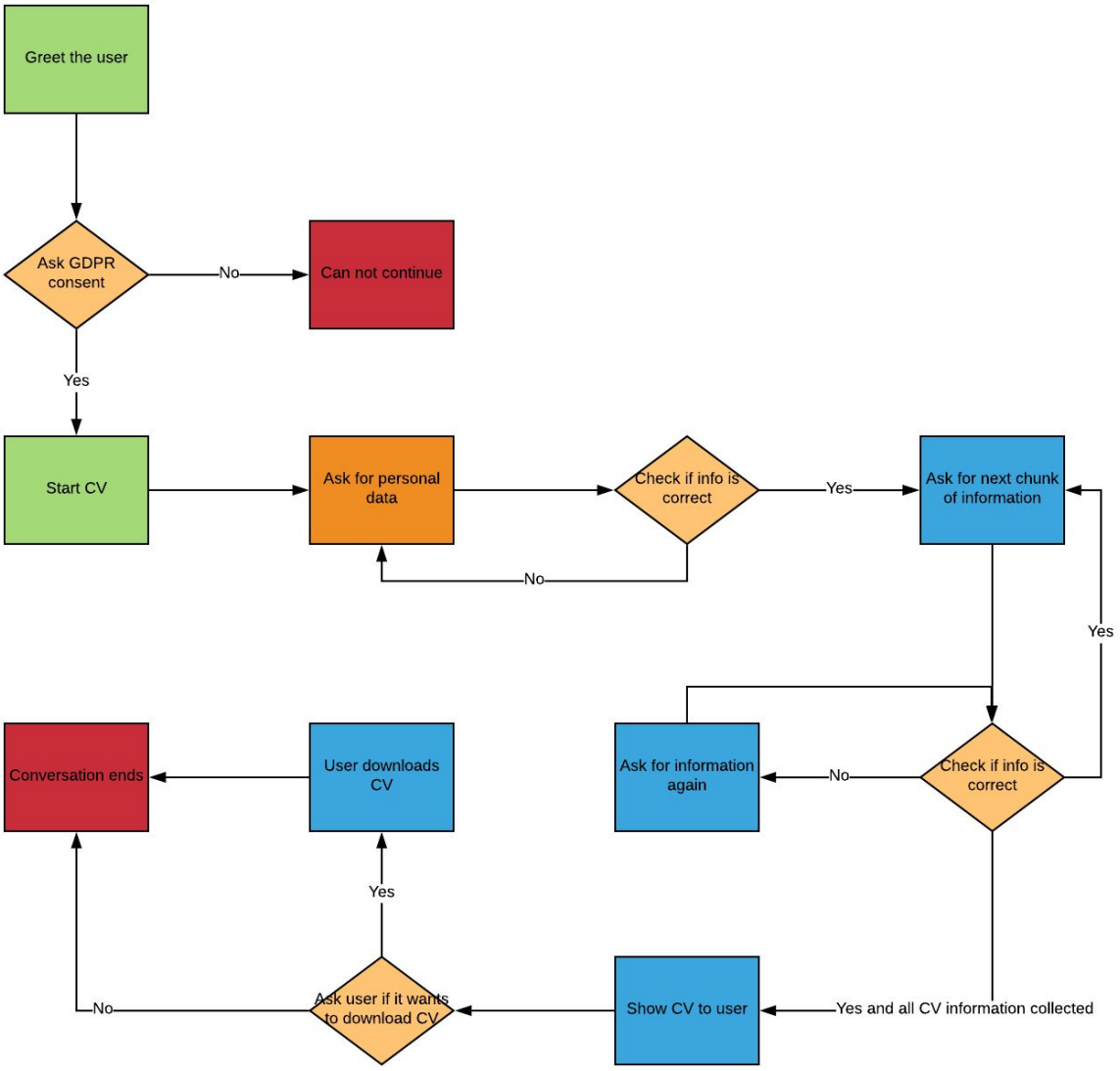

Figure 4.7: Conversation flow.

Firstly the chatbot greets the user, and when trying to start building a CV it asks for consent over data provided, explaining what will data be used for and how to delete it if the user wants to. After that, the CV building starts and the bot asks for information and checks it to ensure it is correctly added. If it it no it asks for it again and repeats the process for every section. Once the CV is complete, it asks the user if it wants to download the CV. Once the CV download is sorted out, the conversation ends.

This figure describes the general diagram flow of how the app should work:

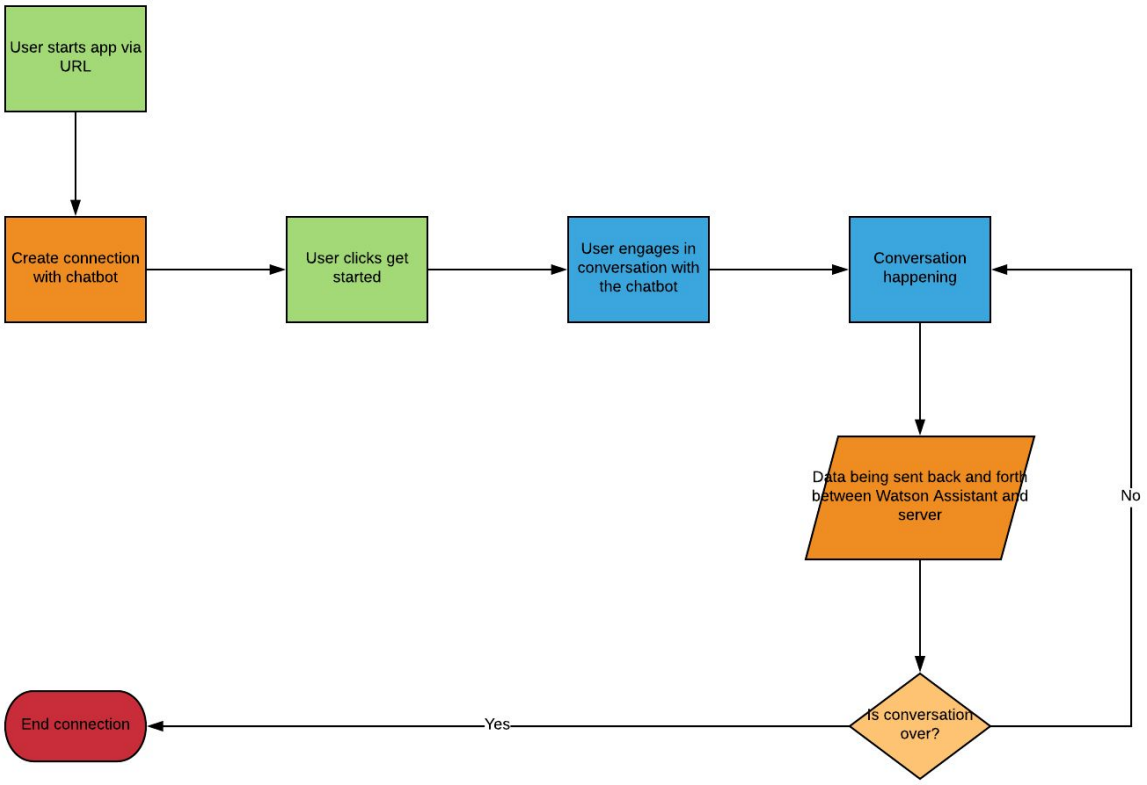

Figure 4.8: System diagram flow

The user connects to the app via the url provided. The server connects to the chatbot and the user can start chatting with it. The user types in the interface provided to chat with the bot and the data gets prepared in a package and sent to the chatbot via an API call, then its processed and an answer is sent back to the user. Once the conversation is over the connection is ended.

Here is an overview of the data package structure:

```
var payload = {
    workspace_id: workspace,
    context: req.body.context || {},
    input: req.body.input || {}
};
```

Listing 4.9: Package structure.

The package sent to Watson Assistant is an object "*payload*" which contains the workspace where the service is located, the context variables in case there are any, and the input of the user.

CHAPTER 5

5.- EVALUATION

This section defines the case studies that have been carried out in order to test the system's functionalities. Also the traceability matrix will be shown.

5.1.- Test cases

The case studies will be presented in a table similar to this template:

Case Study	
ID	CS-XY
Objective	Objective of the case
Precondition	Precondition for the case
Description	Description of execution
Outcome	Result of the case
Status	Status of the case
Traceability	Traceability to requirements

Table 5.1: Case study template.

Each field of the template is defined as follows:

- ID: Unique ID that identifies the case study.
- Objective: Objective to be reached by the case study.
- Description: Brief description of the execution for the case study.
- Outcome: Result after executing the steps described in the description.
- Status: Reflects if the case study has been carried out or not. Can be "*Verified*" or "*Not verified*".
- Traceability: Describes the requirements that have been tested in this case study.

Case Study	
ID	CS-01
Objective	Test app domain
Precondition	App must be hosted in IBM Cloud.
Description	The url will be accessed to see if it connects to the app.
Outcome	The user is able to connect to the app.
Status	Verified
Traceability	UR-01, FR-01

Table 5.2: Case study 01.

Case Study	
ID	CS-02
Objective	Test access to chatbot
Precondition	There must be a user interface.
Description	The user clicks the get started button to access the chatbot.
Outcome	The user gains access to the chatbot.
Status	Verified
Traceability	UR-02, FR-02

Table 5.3: Case study 02.

Case Study	
ID	CS-03
Objective	Test conversation
Precondition	The chatbot must have an interface and be connected to Watson Assistant.
Description	The user types something in the chatbot interface.
Outcome	The user is able to send messages to the chatbot and receive them.
Status	Verified
Traceability	UR-03, FR-03, FR-04

Table 5.4: Case study 03.

Case Study	
ID	CS-04
Objective	Test conversation flow
Precondition	The chatbot must have an interface and be connected to Watson Assistant.
Description	The user is guided through the conversation by the chatbot to create a CV according to the nodes that have been laid out for the conversation flow.
Outcome	The user is guided through the conversation while creating a CV.
Status	Verified
Traceability	UR-03, UR-04, FR-03, FR-04, FR-05

Table 5.5: Case study 04.

Case Study	
ID	CS-05
Objective	Test CV save
Precondition	The user must have built a CV using the chatbot
Description	The chatbot shows a copy of the CV built to the user so it can be saved by the user.
Outcome	The user saves a copy of the CV in the PC.
Status	Verified
Traceability	UR-05, FR-06

Table 5.6: Case study 05.

Case Study	
ID	CS-06
Objective	Test end connection
Precondition	The user must have finished talking with the chatbot..
Description	The user finishes the session by saying goodbye to the chatbot.
Outcome	The session is ended.
Status	Verified
Traceability	UR-06, FR-07

Table 5.7: Case study 06.

Case Study	
ID	CS-07
Objective	Test context
Precondition	The user must be chatting with the chatbot
Description	The context variable are saved to be sent to the Watson Assistant service.
Outcome	The context variable are saved and sent to the Watson Assistant service.
Status	Verified
Traceability	FR-08

Table 5.8: Case study 07.

Case Study	
ID	CS-08
Objective	Test coding
Precondition	The code must be written down.
Description	Check the code.
Outcome	The code has been checked and is written in the required languages.
Status	Verified
Traceability	NFR-01

Table 5.9: Case study 08.

5.2.- Traceability matrix

	CS-01	CS-02	CS-03	CS-04	CS-05	CS-06	CS-07	CS-08
UR-01	X							
UR-02		X						
UR-03			X	X				
UR-04				X				
UR-05					X			
UR-06						X		
FR-01	X							
FR-02		X						
FR-03			X	X				
FR-04			X	X				
FR-05				X				
FR-06					X			
FR-07						X		
FR-08							X	
NFR-01								X

Table 5.10: Traceability matrix.

CHAPTER 6

6.- LEGAL FRAMEWORK AND SOCIO-ECONOMIC IMPACT

6.1.- Legal Framework

This section exposes the legal regulations that are relevant within this project.

6.1.1.- Applicable Legislation

Since this project uses personal data from the user to build the CV the GDPR (*General Data Protection Regulation*) applies to it. Even though the application manages personal information, there is no sensitive personal information from the user. For the application to be GDPR compliant it needs to ask the user for consent on the use of the data. To ask for it, it has to state which is the use it is going to give to the data and where it is going to be stored, as well as how long it will be retained. The user has to have access to information at all time and be able to delete it or ask for it to be deleted whenever the user sees fit. Also, personal information and sensitive personal information must be encrypted and the user must be notified if any data breach may affect the data provided. All this according to according to *GDPR EU 2016/679* regulation approved on April 14th 2016.[9] For this purpose, the chatbot asks for consent on data before building the CV, and this data will be deleted once the user finishes the session. As well, all data that goes through IBM Cloud (packages sent to the chatbot) is by default encrypted.

6.1.2.- Intellectual property

All libraries used are Open-source. They each come in a file with their respective license. This allows the use, creation of derived work, reproduction and distribution as long as the libraries remain in their separate own files with their respective license when any of these actions are given.

IBM holds copyright over Watson Assistant and IBM Cloud. The code for the structure of the chatbot has an Apache License which grants a copyright and patent license over it which allow the use, preparation of derivative work, reproduction and distribution. However distribution needs to meet the following requirements:

- A copy of the license must be given to the recipients of any of this or derived work.
- It must reflect, if files are modified, notifications about its modification
- All copyright, patent, trademark and attribution notices related to any of the derivative work must be retained.

- If a "NOTICE" file is present in the distribution, then all distributed copies must contain an accessible and readable copy of the attribution notices. The purpose of this file is only informational and does not modify the license. An own "NOTICE" file can be added. [10]

6.2.- Socio-Economic Impact

Regarding the social part, this project can make easier the task of making a CV, helping out people that are creating their first CV or maybe updating their existing one. For example, this project can be implemented within the "*UC3M Orientación y Empleo*" service to help students create a CV for possible internships, and maybe make a CV for a future job updating the internship CV.

As for the economic impact, it would be an indirect impact, since the application would be an influence in the process of obtaining a job for the people using it, improving the chances of obtaining said job.

6.2.1.- Budget

This section will display the costs of developing this project, divided into direct costs and indirect costs. The project has an expected development time of 4 months.

The programmer will fill the tasks of programming and testing the code. The designer will analyze the requirements and design the conversation according to them.

The salaries were obtained from the BOE ("*Boletín Oficial del Estado*")[11] for the year 2018.

The direct costs are for staff in the following table:

Staff	Salary/Month	Months	Final cost
Programmer	1,438.08€	4	5,752.32€
Designer	1,438.08€	2	2,876.16€
Project manager	1,089.20€	4	4,356.80€
TOTAL			12,985.28€

Table 6.1: Staff direct costs

and equipment in the next table:

Asset	Unit cost	Units	Final cost
Laptop	999.99€	2	1,999.98€
Mouse	14.95€	2	29.90€
TOTAL			2,029.88€

Table 6.2: Equipment direct costs

Indirect costs are stated in the following table:

Service	Cost/Month	Months	Final cost
Watson Assistant	31.25€	4	125€
Energy costs	106.55€	4	426.20€
Internet connection	24.95€	4	99.80€
TOTAL			651€

Table 6.3: Indirect costs

The costs of Watson Assistant are 0.0025€ per API call. Each time a package of information is sent to the Watson assistant API an API call is generated. Considering the nodes from our conversation flow, an average of 50 accesses are done per session since the conversation is guided and quite linear, and the average number of sessions per month is 250, the costs are therefore the accesses times the average of sessions per month, giving a total of 31.25€.

Adding all costs, a total of 15,666.16€ is obtained. If we assume that a revenue of a 10%, a total of 1,566.61€ will be obtained from this project.

CHAPTER 7

7.- CONCLUSIONS AND FUTURE WORK

This chapter gathers the conclusions that have been come to from making this project, as well as future work that can be done regarding the project.

7.1.- Conclusions

For this project, the objectives stated in the introduction were achieved, but if there was more time availability more functionalities could have been added. The most important part within the project was the research about the tools that were going to be used, as they were all new and had different capabilities. While researching the tools some ideas changed regarding the capabilities of said tools to adjust what could be done with what wanted to be done. Also, at the start, building this project at a very small scale seemed quite easy, but as more nodes were added to the conversation and more functionalities wanted to be added, the complexity and the difficulty started growing. Building a simple bot that gives always the same answer and has a fixed and small conversation is easy, but building an interactive chatbot that allows the user to feel like it is talking to a human and that makes the user feel comfortable is quite a challenge. The interface for the chatbot is not as intuitive to build as it would seem, although the tool UI in IBM Cloud is quite simple. Not sticking just to the cognitive part, but adding a bit of AI on top of what is already built could add more flexibility to the bot as well as learning for better interaction with the users. The bot that has been built is a simple solution for building a CV that can be improved and has a lot of applications in the real world with a bit of polishing.

In conclusion, a lot was learnt from different new tools and from different areas, and despite the limitation from the tools and time that could be spent on the project something that could be the seed of a big application was created.

7.2.- Future Work

As future work to complement the project done, other tasks can be carried out, such as:

· Polishing the conversation to add better guidance or digression from the main conversation if the user has doubts about what to put into the CV for example.

· Adding "*chit-chat*" handling for feedback or offtopic questions, adding a functionality to learn from questions that the chatbot can not handle yet and are somewhat general, such as "What time is it?" and other offtopic questions. Also the feedback handling

would help the chatbot use better responses or change functionalities based on said feedback.

· Adding different types of CV templates in order to apply to different types of jobs with a more prepared CV regarding that job's area.

· Implementing a functionality to search for information about different companies where the user might want to apply to, allowing the user to make a better CV. This could be done by implementing Watson Discovery which helps to recover information from big amount of data and can give the user information that has been gathered about companies.

· Instead of just sticking to the cognitive part, some machine learning could be added to add some learning to the bot in order to make the conversation experience for the user better and more human-like.

APPENDIX A

USER MANUAL

The following steps show how to setup and run the project:

1.- Download and install Node.js and NPM. (NPM will be installed with Node.js so there is no need to download it separately).

2.- Navigate to the folder where the project is contained.

3.- Run the command "*npm install*" to download and install all the modules needed for the project to work. All the modules needed are stated on the JSON file "*package.json"*.

4.- Run the command "*npm start*" or "*node app.js*" to start executing the code.

5.- A message with the port in which the code is running should be prompted in the console:

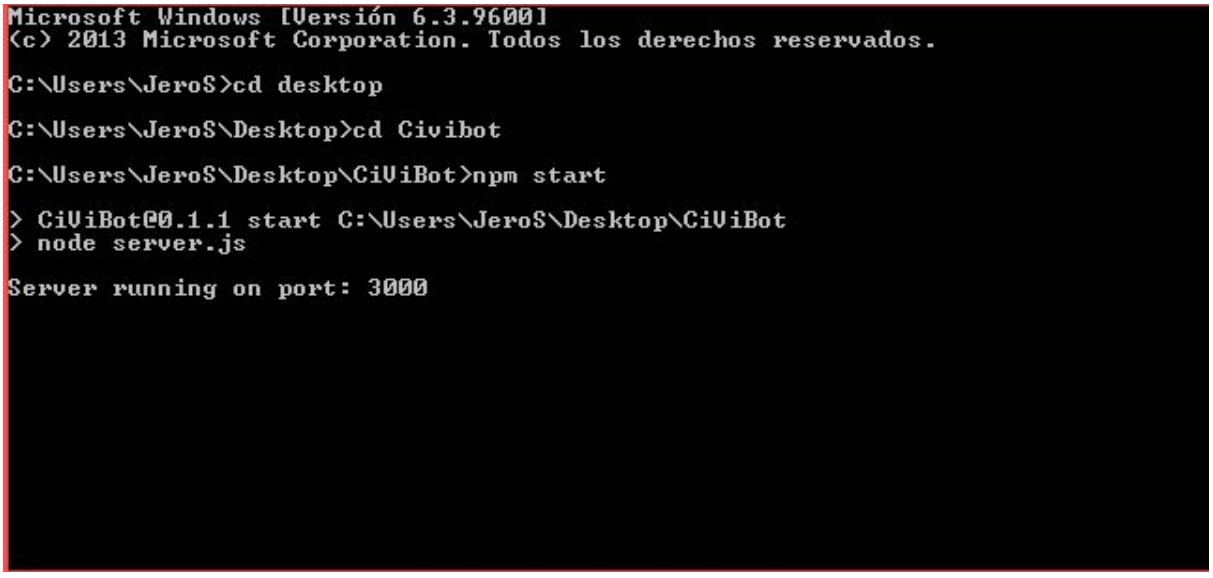

Figure A.1: Execution sample.

BIBLIOGRAPHY

[1] F. M. Yubal, "Así era ELIZA, el primer bot conversacional de la historia", *Xataka*, 27-05-2017. **[En línea]. Disponible en:** https://www.xataka.com/historia-tecnologica/asi-era-eliza-el-primer-bot-conversacional-de-la-historia.

[2] Casper, "insomnobot-3000", *Insomnobot-3000*, 21-09-2016. **[En línea]. Disponible en:** http://insomnobot3000.com/

[3] A. Rao, "Poncho the weather cat, an interview with Greg Leuch"", *Chatbots Magazine*, 24-03-2017. **[En línea]. Disponible en:** https://chatbotsmagazine.com/poncho-the-weather-cat-an-interview-with-greg-leuch-908cfd827aba

[4] J. Vincent, "Baidu launches medical chatbot to help Chinese doctors diagnose patients", *The Verge*, Fecha de publicación. **[En línea]. Disponible en:** https://www.theverge.com/2016/10/11/13240434/baidu-medical-chatbot-china-melody

[5] H. Saeed, "RightClick.io Uses AI-Powered Chatbot to Create a Website", *Chatbots Magazine*, 09-10-2016. **[En línea]. Disponible en:** https://chatbotsmagazine.com/rightclick-io-ai-website-builder-82f0e6f3c61c

[6] P. Moya, "Habla con Mitsuku, el bot que no podrás distinguir de un humano", *Omicrono*, 18-10-2016. **[En línea]. Disponible en:** https://omicrono.elespanol.com/2016/10/mitsuku-bot-conversacional/

[7] B. McLaughlin, *Whats is Node.js?*, 1st ed. O'Reilly Media Inc, 2011. **[En línea]. Disponible en:** https://play.google.com/books/reader?id=SRQHcjljUtwC&hl=es&pg=GBS.PT1

[8] P. Teixeira, *Professional Node.js: Building Javascript Based Scalable Software*, 1st ed. John Wiley & Sons, 2012. **[En línea]. Disponible en:** https://books.google.es/books?hl=es&lr=&id=ZH6bpbcrlvYC&oi=fnd&pg=PR27&dq=nodejs&ots=mOCv8DmkHe&sig=E-nIGWNDy8HWEuw4j8_ZYccsE7g#v=onepage&q&f=false

[9] M. Schulz, J.A. Hennis-Plasschaert, "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL", European Union, Belgium, **Technical report** (EU) 2016/679, 2016

[10] apache.org, "Apache License V2.0"

[11] E. Ministerio de Empleo y Seguridad Social**,** "MINISTERIO DE EMPLEO Y SEGURIDAD SOCIAL", BOE, Spain, **Technical report** 3156-58, 2018.